Note on Changes to Simkit in Version 1.2.14

1. Changes from Version 1.2.13

The two most significant changes are in the behavior of RandomVariateFactory in creatingt RandomVariate instances and the addition of dynamic properties.

1.1. RandomVariateFactory uses One Instance of MersenneTwister

RandomVariateFactory now uses the MersenneTwister as the default RandomNumber type. Also, by default, a single instance is used to give each RandomVariate instance its UN(0,1) random numbers. This makes it easier to avoid spur ious correlation in the beginning without having the novice user to have to worry about setting seeds. If correlation induction is desired, or if for whatever reason the user wants to maintain several instances of RandomNumber or manage the seeds herself, that is possible as well, with about as much work as previously required.

The bottom line is that the user can simply get references to RandomVariates by invoking RandomVariateFactory.getInstance() and be assured that the resulting generated variates will be independent (at least as independent as the underlying RandomNumber instance). If another RandomNumber type is desired, invoke RandomVariateFactory.setDefaultRandomNumber(RandomNumber), passing the desired instance of RandomNumber. Henceforth, that instance will be used for all subsequent calls to RandomVariateFactory.getInstance(). However, all RandomVariates created before that will have their original RandomNumber instances. They can be switched to the new one by invoking setRandomNumber() on each instance. Since this is tedious and potentially risky (some might be missed), it is advised that this be done before any RandomVariate instances are created from RandomVariateFactory.

1.2. Dynamic Properties

All subsets of BasicSimEntity (which include SimEntityBase) have the capability of creating dynamic properties on-the-fly using setProperty(String, Object) and getProperty(String).

When setProperty() is invoked, the object's class is checked to see if there is a static property of that name. If so, then it is attempted to be set, throwing an exception if the type of object is incompatible. If there is no static property of that name, then the property is added with the passed in value as a dynamic property. This is done with a PropertyChangeDispatcher delegate, in which all the real work is done. Any previously defined dynamic property has its value overwritten.

When getProperty(String) is invoked, the object again first checks to see if there is a staic property of that name. If so, then its value is obtained and returned. If the property is primitive, the return is wrapped in the corresponding wrapper class. If no static property is found, then the object attempts to get the value of a dynamic property of that name. If found, the value is returned. If not found, the value null is returned.

Normally the toString() for BasicSimEntity (and below) only lists the parameters (i.e. statica properties with both getter and setter methods) and omits state variables and dynamic properties.

However, if the property justDefinedProperties is set to false, then all dynamic properties (as well as state variables) are included in the toString().

A list of all dynamically defined properties can be obtained as a String[] array using getAddedProperties().

1.3. Array Properties in toString()

Version 1.2.13 introduced a default toString() that listed all the parameters and their values. However, an array property was printed using the array toString(), producing un-useful output. Version 1.2.14 prints out the name, index, and value of each 1-dimensional array element. For 2-dimension (and higher) arrays, it reverts to the array toString(). Perhaps a future version will have clever recursive code that prints arrays of any dimension.

1.4. CollectionSizeTimeVaryingStats

A new SimpleStats class has been added to the simkit.stat package, CollectionSizeTimeVaryingStats. An instance of this class listens for a property change of a given name of type java.util.Collection. When the change is heard, the size of the Collection is used to populate a time-varying statistical counter.

2. Changes for Version 1.2.13

The primary changes for version 1.2.13 were the possibility of having more than one Event List and a generated toString() for BasicSimEntity (and below) that listed the values of each property. Normally, only the parameters are listed, not state variables. This is done using introspection, with only properties with both setters and getters (state variables should not have public setter methods).

2.1. Multiple Event Lists

Most users are unaffected by this, but when using Simkit as the basis for a Web Service, it becomes desirable to have multiple simulations running independently. So there is the possibility of creating more than one EventList. Each created Event List has a unique id associated with it, an integer. The Schedule class now has a default Event List that it passes through its methods to. By default, Event List "0" is the default, but that can be changed by the user who wishes it.

2.2. Useful toString() for BasicSimEntity and SimEntityBase

The most widely used purpose of a SimEntity's toString() was to ensure that it had properly received the correct parameter values. Therefore, for version 1.2.13 a toString() method was written for BasicSimEntity (that works on SimEntityBase and all subclasses) that lists the parameters and their values, with the SimEntity's name as the first line.. Although it is not the prettiest output in the world, it provides the desired functionality at a zero effort to the user. Any new parameters are automatically included. State variables are omitted, so if they are desired (or if any other functionality is desired), then the user is back to writing their own toString() method. As noted above, in version 1.2.13 array parameters were not handled very well, but in version 1.2.14 single dimensional array parameters are.